The Relative Power of Composite Loop Agreement Tasks

Maurice Herlihy and Vikram Saraph

June 12, 2015

Abstract

Loop agreement is a family of wait-free tasks that includes set agreement and simplex agreement, and was used to prove the undecidability of wait-free solvability of distributed tasks by read/write memory. Herlihy and Rajsbaum defined the algebraic signature of a loop agreement task, which consists of a group and a distinguished element. They used the algebraic signature to characterize the relative power of loop agreement tasks. In particular, they showed that one task implements another exactly when there is a homomorphism between their respective signatures sending one distinguished element to the other. In this paper, we extend the previous result by defining the *composition* of multiple loop agreement tasks to create a new one with the same combined power. We generalize the original algebraic characterization of relative power to compositions of tasks. In this way, we can think of loop agreement tasks in terms of their basic building blocks. We also investigate a category-theoretic perspective of loop agreement by defining a category of loops, showing that the algebraic signature is a functor, and proving that our definition of task composition is the "correct" one, in a categorical sense.

1 Introduction

A *task* is a distributed problem in which each process begins with an input, communicates with others, and returns an output according to the task's specification. Common examples of tasks include consensus [3], set agreement [2], and renaming [1]. *Protocols* are distributed programs that solve tasks. A protocol is *wait-free* if every non-faulty process running the protocol eventually finishes execution, regardless of other process failures. One task *implements* another if a protocol for the first task can be modified in a simple way to solve the second task.

Loop agreement is a family of tasks that models the convergence of processes along a distinguished loop of a given space, and includes simplex agreement and set agreement. One application of loop agreement is a simple proof of the undecidability of solvability of distributed tasks by read/write memory [9]. Herlihy and Rajsbaum defined the *algebraic signature* of a loop agreement task in terms of a group G and an element $g \in G$. They proved that the algebraic signature completely characterizes the relative power of loop agreement

tasks [10], in the following sense. If tasks T_1 and T_2 have signatures (G_1, g_1) and (G_2, g_2), respectively, then T_1 implements T_2 exactly when there is a group homomorphism $\phi : G_1 \to G_2$ mapping g_1 to g_2. Thus the operational problem of loop agreement tasks implementing one another is reduced to an algebraic characterization.

In this paper, we describe how several loop agreement tasks can implement others, define compositions of loop agreement tasks, and extend the aforementioned algebraic characterization to these compositions. Roughly speaking, the *composition* of n loop agreement tasks is a task in which each process solves each of the n tasks in parallel. We show that tasks $\{T_i\}$ with signatures $\{(G_i, g_i)\}$ solve T with signature (G, g) if and only if there is a homomorphism $\phi : G_1 \times \cdots \times G_n \to G$ mapping $(g_1, \ldots, g_n)$ to g. We also provide a means of replacing the loop agreement tasks $\{T_i\}$ with an equivalent task $\prod T_i$, called the *composition* of the $\{T_i\}$. This composition of tasks is also a loop agreement task, and has relative power equivalent to that of all the $\{T_i\}$. That is, the $\{T_i\}$ implement $\prod T_i$ and $\prod T_i$ implements each T_i.

Finally, we take a category-theoretic approach to loop agreement in order to show that we have the correct notion of task composition. We define a category of loop agreement tasks, **Loop**, and show that the map assigning tasks to algebraic signatures is a functor into the category of pointed groups, **pGrp**. We also show that composition of loop agreement tasks is the categorical product in **Loop**, which strongly suggests that composition of tasks as defined in this paper correctly captures the operational meaning of parallel composition. We believe this category-theoretic approach may inspire future work on parallel composition of more general tasks beyond loop agreement, and may also inspire other work on applying category theory to general tasks.

Section 2 describes related work. Section 3 is a whirlwind tour of distributed tasks, algebraic topology, and loop agreement. Section 4 defines multiple implementation and composition of tasks and proves the main theorem. Section 5 provides an informal introduction to category theory and describes the category-theoretic view of loop agreement. Section 6 presents simple applications of our results, and in Section 7 we conclude with ideas for possible future work.

2 Related Work

Herlihy and Shavit introduced the use of algebraic topology [11, 12], and in particular, homology theory to prove various impossibility results pertaining to set agreement and renaming. Since then homology theory has been used to prove other impossibility results in distributed computing [7, 8]. Gafni and Koutsoupias were the first to use the fundamental group in understanding distributed tasks [4] by showing the undecidability of wait-free solvability of certain tasks. Herlihy and Rajsbaum obtained similar undecidability results in other models which include loop agreement [9], and also characterized the relative power of loop agreement tasks via their algebraic signatures [10].

Loop agreement has also been generalized to higher dimensions. Liu, Xu, and Pan define *n-rendezvous tasks* [16], where processes begin on distinguished vertices of an embedded $(n-1)$-sphere of an n-dimensional complex, and converge on a simplex of the embedded sphere. They generalize the algebraic signature characterization to a subclass of rendezvous

tasks called *nice* rendezvous tasks, which are tasks whose output complexes have trivial homology groups below and above dimension n, and a free Abelian n-th homology group. The authors apply their main result to show there are countably infinite inequivalent nice rendezvous tasks.

Liu, Pu, and Pan explore a lower-dimensional variant of loop agreement called *degenerate loop agreement* [15], which unlike loop agreement includes binary consensus. Processes begin on a 1-dimensional complex, or a graph, and must converge to one of two possible starting locations in the graph. The authors prove that there are only two inequivalent tasks degenerate tasks: the trivial task and binary consensus.

3 Background

In the first subsection, we describe the mathematical model used for distributed tasks, of which more details can be found in Herlihy, Kozlov, and Rajsbaum [6]. In the second subsection, we summarize important definitions and results from algebraic topology.

3.1 Distributed Computing

Formally, a (colorless) *task* is a triple $(\mathcal{I}, \mathcal{O}, \Gamma)$, where objects $\mathcal{I}$ and $\mathcal{O}$, called the *input* and *output complexes* of the task, are mathematical structures known as simplicial complexes. A *simplicial complex* on a set V is a collection of subsets $\mathcal{C}$ of V such that $\mathcal{C}$ is downward closed under the subset relation. Complexes can be thought of as higher-dimensional graphs where "edges" may "connect" more than two vertices. In the context of tasks, vertices of $\mathcal{I}$ represent process input values, while simplexes of $\mathcal{I}$ represent valid input combinations. Likewise, vertices of $\mathcal{O}$ represent process output (or decision) values, and simplexes represent valid output combinations. Relating $\mathcal{I}$ and $\mathcal{O}$ is the map $\Gamma : \mathcal{I} \to 2^{\mathcal{O}}$, which is called the task's *specification map*, and carries simplexes of $\mathcal{I}$ to subcomplexes of $\mathcal{O}$ in a monotonic way[1]. The map Γ associates each input combination with a set of legal output combinations.

Protocols are objects that solve tasks, and are also modeled by triples $(\mathcal{I}, \mathcal{P}, \Xi)$. As with tasks, $\mathcal{I}$ is the protocol's *input complex*. The object $\mathcal{P}$ is also a simplicial complex, which is called the *protocol complex*, and is similar to a task's output complex, but has a slightly different meaning. Rather than a final decision value, a vertex in $\mathcal{P}$ represents a process's uninterpreted state (or view) after running the protocol. The map $\Xi : \mathcal{I} \to 2^{\mathcal{P}}$, called the *execution map*, is monotonic, and represents the possible states in which processes may result after running the protocol.

A *simplicial map* $\delta : \mathcal{I} \to \mathcal{O}$ between two complexes is a vertex map that send simplexes to simplexes; that is, $\delta(\sigma) \in \mathcal{O}$ for each $\sigma \in \mathcal{I}$. A protocol $(\mathcal{I}, \mathcal{P}, \Xi)$ *solves* $(\mathcal{I}, \mathcal{O}, \Gamma)$ if there exists a simplicial map $\delta : \mathcal{O} \to \mathcal{P}$, called a *decision map*, that respects the task specification Γ. Formally, δ respects Γ if for each simplex $\sigma \in \mathcal{I}$, we have $(\delta \circ \Xi)(\sigma) \subseteq \Gamma(\sigma)$.

Some tasks are inherently harder than others, and sometimes we can transform a protocol for one task into a protocol for another. We say task T_1 *implements* T_2 if we can use the output complex of T_1 (or a subdivision of it) as a protocol complex for solving T_2.

[1] In general, if $\mathcal{A}$ and $\mathcal{B}$ are simplicial complexes, then a function $\Phi : \mathcal{A} \to 2^{\mathcal{B}}$ is called a *carrier map* if for each $\sigma \subseteq \tau \in \mathcal{A}$, $\Phi(\sigma)$ is a simplicial complex, and $\Phi(\sigma) \subseteq \Phi(\tau)$ (or Φ is *monotonic*).

Mathematically speaking, if $T_1 = (\mathcal{I}, \mathcal{O}_1, \Gamma_1)$ and $T_2 = (\mathcal{I}, \mathcal{O}_2, \Gamma_2)$, then T_1 implements T_2 if there exists an natural number N and a simplicial map $\phi : \text{Bary}^N(\mathcal{O}_1) \to \mathcal{O}_2$ such that $(\phi \circ \text{Bary}^N \circ \Gamma_1)(\sigma) \subseteq \Gamma_2(\sigma)$ for each $\sigma \in \mathcal{I}$. The barycentric subdivision operator Bary is a topological operator (see the next section) that models read/write memory. Two tasks are *equivalent* if they implement each other.

3.2 Algebraic Topology

Before we can define loop agreement, we must briefly introduce the relevant machinery from algebraic topology. We assume a basic understanding of point-set topology. The algebraic topology used is at the undergraduate level, of which a formal treatment can be found in Hatcher [5]. We begin with the formal definition of a simplicial complex.

3.2.1 Simplicial Complexes

Definition 3.1. Let V be any set, whose elements are called *vertices*. A *simplicial complex* (over V) is a set of subsets $\mathcal{C}$ of V such that for each set $\tau \in \mathcal{C}$, if $\sigma \subseteq \tau$, then $\sigma \in \mathcal{C}$. That is, $\mathcal{C}$ is downward closed under taking subsets. Elements of $\mathcal{C}$ are called *simplexes*.

We can think of simplicial complexes as a generalization of graphs, where simplexes may be incident to more than two vertices. Graphs are then precisely the simplicial complexes whose simplexes contain at most two vertices. Nontrivial graphs have dimension 1, and in general, the *dimension* of a complex $\mathcal{C}$ is $n-1$, where n is the size of the largest simplex in $\mathcal{C}$. The dimension of a simplex σ is simply $|\sigma| - 1$. The *standard n-simplex*, Δ^n, is the simplicial complex on $n+1$ vertices containing all possible simplexes. By convention, we will use $\{0, \ldots, n\}$ for the vertex set of Δ^n.

A *subcomplex* of $\mathcal{C}$ is a subset $\mathcal{B} \subseteq \mathcal{C}$ that is also a simplicial complex. For each nonnegative integer k, the *k-skeleton* of $\mathcal{C}$, denoted $\text{skel}^k(\mathcal{C})$, is the subcomplex of $\mathcal{C}$ containing all simplexes of dimension at most k.

The above formulation of simplicial complexes defines them in a purely combinatorial way, but complexes can also be realized as topological spaces. Notationally, if $\mathcal{C}$ is a complex, then its geometric realization is denoted by $|\mathcal{C}|$. As previously mentioned, the *barycentric subdivision* is an operator that models read/write memory, and is better understood geometrically than combinatorially. Given a geometric simplicial complex $|\mathcal{C}|$, we can create another geometric simplicial complex by adding new vertices to the barycenter of each simplex, and adding new simplexes accordingly. This gives rise to an abstract simplicial complex, denoted $\text{Bary}(\mathcal{C})$. Notice that the barycentric subdivision does not change the geometric realization of the original complex; that is, $|\text{Bary}(\mathcal{C})| = |\mathcal{C}|$.

The barycentric subdivision is also an important tool in approximating continuous functions with simplicial maps. If $f : |\mathcal{A}| \to |\mathcal{B}|$ is a continuous function between complexes, then a simplicial map $\phi : \mathcal{A} \to \mathcal{B}$ is called a *simplicial approximation* of f if for every $p \in |\mathcal{A}|$, $|\phi|(p)$ is contained in the smallest simplex containing $f(p)$. Using the barycentric subdivision, we can construct a simplicial approximation of any continuous function, as stated below.

Fact 3.2 (Simplicial Approximation)**.** *Let $f : |\mathcal{A}| \to |\mathcal{B}|$ be a continuous function between simplicial complexes. Then there exists an $N \in \mathbb{N}$ and a simplicial map $\phi : Bary^N(\mathcal{A}) \to \mathcal{B}$*

that is a simplicial approximation of f.

We can take products of simplicial complexes. The product of two complexes is another complex that combines the structures of the original two.

Definition 3.3. Let $\mathcal{C}_1$ and $\mathcal{C}_2$ be simplicial complexes, and let $V(\mathcal{C}_1)$ and $V(\mathcal{C}_2)$ be their vertex sets, respectively. Then the *(categorical) product of simplicial complexes* is a complex $\mathcal{C}_1 \times \mathcal{C}_2$ with vertex set $V(\mathcal{C}_1) \times V(\mathcal{C}_2)$. A subset σ of $V(\mathcal{C}_1) \times V(\mathcal{C}_2)$ is a simplex in $\mathcal{C}_1 \times \mathcal{C}_2$ if and only if $\rho_1(\sigma)$ and $\rho_1(\sigma)$ are simplexes in $\mathcal{C}_1$ and $\mathcal{C}_2$, where ρ_1 and ρ_2 are projections onto the first and second coordinates, respectively.

Intuitively, the product of complexes is a way of combining two complexes in the "best possible way," and operationally, the product captures all possible combinations of process views if two tasks are solved in parallel. It is an important technical point that the product of complexes and product of topological spaces are not the same; it is not true that $|\mathcal{A}| \times |\mathcal{B}|$ and $|\mathcal{A} \times \mathcal{B}|$ are homeomorphic. They are, however, "homotopy equivalent," which is a type of equivalence described in the next section.

To each topological space we can assign an invariant called the fundamental group, a basic construct taken from algebraic topology. The fundamental group is used to define the algebraic signature of a loop agreement task.

3.2.2 Homotopy and the Fundamental Group

Given a topological space X and a basepoint $x_0 \in X$, a *loop* in X based at x_0 is a continuous function $\lambda : [0,1] \to X$ such that $\lambda(0) = \lambda(1) = x_0$. Two loops λ_1 and λ_2 based x_0 are (loop) *homotopic* if one loop can be continuously deformed to the other. More precisely, λ_1 and λ_2 are homotopic if there is a continuous function $H : [0,1] \times [0,1] \to X$ such that $H(0,-) = \lambda_1$, $H(1,-) = \lambda_2$, and $H(-,0) = H(-,1) = x_0$. Homotopy is an equivalence relation. We write $[\lambda]$ to denote the equivalence class of all loops homotopic to λ.

Let $\alpha : [0,1] \to X$ and $\beta : [0,1] \to X$ be two loops based at x_0. Then we can *concatenate* α and β to get another loop, $\alpha \cdot \beta$, defined by traversing α, returning to x_0, and then traversing β. The loop $\alpha \cdot \beta : [0,1] \to X$, also based at x_0, is defined as

$$(\alpha \cdot \beta)(t) = \begin{cases} \alpha(2t) & \text{for } 0 \le t \le \frac{1}{2} \\ \beta(2t-1) & \text{for } \frac{1}{2} \le t \le 1 \end{cases}$$

Concatenation behaves well with homotopy. If α and β are homotopic to α' and β', respectively, then $[\alpha \cdot \beta] = [\alpha' \cdot \beta']$. From this it follows that concatenation is associative on classes of loops based at x_0. In fact, concatenation is a group operation on classes of loops based at x_0, with the inverse computed by traversing a loop in the opposite direction, and the identity element being the class of all loops homotopic to the constant loop at x_0. Formally, the inverse of $[\alpha]$ is the class of the loop $\alpha^{-1}(t) = \alpha(1-t)$, and the class $[e]$ of loop $e(t) = x_0$ serves as the identity.

Definition 3.4. Let X be a topological space, and let $x_0 \in X$ be a basepoint. Then the *fundamental group* of X at x_0, denoted $\pi_1(X, x_0)$, is the set of all loop homotopy classes with concatenation as its group operation. If X is path-connected, then $\pi_1(X, x_0)$ is independent of x_0, and we simply write $\pi_1(X)$.

If $f : (X, x_0) \to (Y, y_0)$ is a basepoint-preserving continuous function, then π_1 also induces a group homomorphism $f_* : \pi(X, x_0) \to (Y, y_0)$ called the *induced homomorphism*, defined by $f_*([\lambda]) = [f \circ \lambda]$.

Henceforth, we assume all topological spaces and simplicial complexes under consideration are path-connected. For brevity, if $\mathcal{C}$ is a complex, we write $\pi_1(\mathcal{C})$ instead of $\pi_1(|\mathcal{C}|)$. An important property of the fundamental group is how it behaves with the product of topological spaces.

Fact 3.5. *Let X and Y be topological spaces. Then $\pi_1(X \times Y) \cong \pi_1(X) \times \pi_1(Y)$.*

Homotopy is defined for loops, but it is more generally defined for continuous functions where the domain may not be $[0, 1]$. Two continuous functions $f, g : X \to Y$ are *homotopic* if there is a continuous $H : X \times [0, 1] \to Y$ such that $H(-, 0) = f$ and $H(-, 1) = g$. We write $f \simeq g$ if this is the case. If in addition $X \subseteq Y$ and H fixes X, then H is called a *deformation retraction* and we say Y *deformation retracts* onto X. If δ is a simplicial approximation of a continuous function h, then it is known that $|\delta| \simeq h$.

Using homotopy, we can define a weak equivalence between topological spaces called homotopy equivalence.

Definition 3.6. Let X and Y be topological spaces. Then X and Y are *homotopy equivalent*, or $X \simeq Y$, if there are continuous functions $f : X \to Y$ and $g : Y \to X$ such that $g \circ f \simeq \mathrm{id}_X$ and $f \circ g \simeq \mathrm{id}_Y$. The maps f and g are called *homotopy equivalences* and are *homotopy inverses* of one another.

Homeomorphic spaces are clearly homotopy equivalent. Homotopy equivalent spaces have the same fundamental group.

Fact 3.7. *Let X and Y be topological spaces. If $X \simeq Y$, then $\pi_1(X) \cong \pi_1(Y)$.*

The next few facts are specifically about simplicial complexes. Recall that given two simplicial complexes $\mathcal{A}$ and $\mathcal{B}$, $|\mathcal{A}| \times |\mathcal{B}|$ and $|\mathcal{A} \times \mathcal{B}|$ are not topologically equivalent, though they are homotopy equivalent. See Kozlov's book on combinatorial algebraic topology for a detailed proof of this result [13].

Fact 3.8. *Let $\mathcal{A}$ and $\mathcal{B}$ be simplicial complexes. Then $|\mathcal{A}| \times |\mathcal{B}| \simeq |\mathcal{A} \times \mathcal{B}|$.*

It follows that $|\mathcal{A}| \times |\mathcal{B}|$ and $|\mathcal{A} \times \mathcal{B}|$ have the same fundamental group. This will allow us to pass between the categorical product of $\mathcal{A}$ and $\mathcal{B}$ and the topological product of $|\mathcal{A}|$ and $|\mathcal{B}|$. We will require one more fact relating the fundamental group and the 2-skeleton.

Fact 3.9. *Let $\mathcal{C}$ be a complex. Then the inclusion $\iota : skel^2(\mathcal{C}) \to \mathcal{C}$ induces an isomorphism on fundamental groups.*

This fact can be derived from the following, more general result, which can be found in Hatcher [5]. We call a continuous function $g : |\mathcal{A}| \to |\mathcal{B}|$ *cellular* if g maps skeleta to skeleta, or more precisely, if $g(|\mathrm{skel}^n(\mathcal{A})|) \subseteq |\mathrm{skel}^n(\mathcal{B})|$ for every n. Then every continuous $f : |\mathcal{A}| \to |\mathcal{B}|$ is homotopic to such a map g, as seen below.

Fact 3.10 (Cellular Approximation)**.** *Let $f : |\mathcal{A}| \to |\mathcal{B}|$ be a continuous function between simplicial complexes $\mathcal{A}$ and $\mathcal{B}$. Then f is homotopic to a cellular function $g : |\mathcal{A}| \to |\mathcal{B}|$. Furthermore, if $\mathcal{C} \subseteq \mathcal{A}$ is a subcomplex such that f is already cellular on $|\mathcal{C}|$, then we may require the homotopy between f and g to fix $|\mathcal{C}|$.*

Now suppose we have a homotopy on a subcomplex and we want to extend it to the entire simplicial complex. The next fact, also found in Hatcher [5], allows us to do this.

Fact 3.11 (Homotopy Extension)**.** *Let $\mathcal{C} \subseteq \mathcal{A}$ and $\mathcal{B}$ be simplicial complexes, and let $F : |\mathcal{A}| \to |\mathcal{B}|$ be a continuous function. Suppose we have a homotopy $H : |\mathcal{C}| \times [0,1] \to |\mathcal{B}|$ such that $H(-,0) = F|_{|\mathcal{C}|}$. Then there is a homotopy extending H to all of $|\mathcal{A}|$, respecting F. That is, we can find homotopy $H' : |\mathcal{A}| \times [0,1] \to |\mathcal{B}|$ such that $H'|_{|\mathcal{C}|\times[0,1]} = H$ and $H'(-,0) = F$.*

3.3 Loop Agreement

We need a few more definitions before introducing loop agreement tasks.

Definition 3.12. Let $\mathcal{C}$ be a simplicial complex. An *edge path* in $\mathcal{C}$ is an alternating sequence of vertices and edges, $v_1, e_1, v_2, e_2, \ldots, v_{k-1}, e_{k-1}, v_k$, where $e_i = \{v_i, v_{i+1}\}$. An *edge loop* is an edge path with $v_0 = v_k$.

Definition 3.13. Let $\mathcal{C}$ be a simplicial complex. Then a *triangle loop* in $\mathcal{C}$ is a six-tuple $\lambda = (v_0, v_1, v_2, p_{01}, p_{12}, p_{20})$ such that each v_i is a vertex in $\mathcal{C}$ and p_{ij} is an edge path between v_i and v_j.

Triangle loops are indeed loops in the topological sense, but they can also be viewed as subcomplexes with designated vertices and edge paths. We now have the necessary tools and background to discuss loop agreement tasks. As previously stated, loop agreement is a class of tasks that models convergence of processes on an edge loop of a given space. The precise definition of loop agreement is given below [10].

Definition 3.14. A *loop agreement task* is a task $(\mathcal{I}, \mathcal{O}, \Gamma)$ for which $\mathcal{I}$ is the standard 2-simplex, $\mathcal{O}$ is a (path-connected) 2-dimensional simplicial complex with triangle loop $\lambda = (v_0, v_1, v_2, p_{01}, p_{12}, p_{20})$, and Γ is defined as:

$$\Gamma(\sigma) = \begin{cases} \{v_i\} & : \sigma = \{i\} \\ p_{ij} & : \sigma = \{i, j\} \\ \mathcal{O} & : \sigma = \{0, 1, 2\} \end{cases}$$

Notationally, we write $\mathrm{Loop}(\mathcal{O}, \lambda)$. Input vertices are carried to the designated vertices of λ, the input edges are carried to paths between designated vertices, and the input triangle is carried to the whole output complex. The *algebraic signature* of $\mathrm{Loop}(\mathcal{O}, \lambda)$ is $(\pi_1(\mathcal{O}), \lambda)$, and is used in the main theorem by Herlihy and Rajsbaum [10]:

Theorem 3.15 (Herlihy and Rajsbaum)**.** *Task $\mathrm{Loop}(\mathcal{K}_1, \lambda_1)$ implements $\mathrm{Loop}(\mathcal{K}_2, \lambda_2)$ if and only if there exists a group homomorphism $h : \pi_1(\mathcal{K}_1) \to \pi_1(\mathcal{K}_2)$ such that $h([\lambda_1]) = [\lambda_2]$.*

The main contribution of this paper is parallel composition of tasks and the characterization of their relative power. We allow multiple tasks to implement another, and we generalize the above theorem to multiple tasks. We also show that a loop agreement task being implemented by two others is equivalent to the first being implemented by the composition of the second two.

4 Composite Loop Agreement

4.1 Implementation by Multiple Tasks

Informally, to implement one task by several others, we run protocols for each implementing task and use the combined output as a protocol complex. Given two loop agreement tasks, we will take the product of the output complexes and take the 2-skeleton of the results; this becomes the output complex of the composite task. To obtain a loop in the output complex, we take the "diagonal" of the product of the two original loops. We describe the construction of this loop in more detail.

Definition 4.1. Let $\lambda_1 = (v_0, v_1, v_2, p_{01}, p_{12}, p_{20})$ and $\lambda_2 = (w_0, w_1, w_2, q_{01}, q_{12}, q_{20})$ be triangle loops in complexes $\mathcal{A}$ and $\mathcal{B}$, respectively. Then the *diagonal product* of λ_1 and λ_2, denoted $\lambda_1 \star \lambda_2$, is the triangle loop $(u_0, u_1, u_2, r_{01}, r_{12}, r_{20})$ in $\mathcal{A} \times \mathcal{B}$, where $u_i = (v_i, w_i)$. The path r_{ij} is defined by traversing p_{ij} while w_i is fixed, followed by traversing q_{ij} while v_j is fixed. Note that we will use $p_{ij} \star q_{ij}$ to denote the path defined by r_{ij} as above, though strictly speaking, the $\star$ operator denotes two different operations in $\lambda_1 \star \lambda_2$ and $p_{ij} \star q_{ij}$.

Definition 4.2. Let $T_1 = \mathrm{Loop}(\mathcal{K}_1, \lambda_1)$, $T_2 = \mathrm{Loop}(\mathcal{K}_2, \lambda_2)$, and $T = \mathrm{Loop}(\mathcal{K}, \lambda)$ be loop agreement tasks. Let Γ_1, Γ_2, and Γ be their respective specification maps. We say T_1 and T_2 *implement* T if there is an $N \in \mathbb{N}$ and a simplicial map $\phi : \mathrm{Bary}^N(\mathrm{skel}^2(\mathcal{K}_1 \times \mathcal{K}_2)) \to \mathcal{K}$ such that $(\phi \circ \mathrm{Bary}^N)(\mathrm{skel}^2(\Gamma_1(\sigma) \times \Gamma_2(\sigma))) \subseteq \Gamma(\sigma)$.

Operationally, the participating processes first execute protocols for T_1 and T_2, ending up on a simplex of $\mathcal{K}_1 \times \mathcal{K}_2$. More precisely, because there are at most three participants, they end up on a simplex of $\mathrm{skel}^2(\mathcal{K}_1 \times \mathcal{K}_2)$. They then exchange results via N rounds of reading and writing to "scratchpad" read-write memory, ending up on a simplex of $\mathrm{Bary}^N(\mathrm{skel}^2(\mathcal{K}_1 \times \mathcal{K}_2))$. Finally, each process calls a decision map ϕ to choose a vertex in $\mathcal{K}$.

4.2 Relative Power

In this section we use the following notation for a continuous function that maps one triangle loop to another. If $\mathcal{K}_1$ and $\mathcal{K}_2$ are complexes with triangle loops $\lambda_1 = (v_0, v_1, v_2, p_{01}, p_{12}, p_{20})$ and $\lambda_2 = (w_0, w_1, w_2, q_{01}, q_{12}, q_{20})$, respectively, then we write $f : (\mathcal{K}_1, \lambda_1) \to (\mathcal{K}_2, \lambda_2)$ to denote a continuous function $f : |\mathcal{K}_1| \to |\mathcal{K}_2|$ such that $f(v_i) = w_i$ and $f(|p_{ij}|) \subseteq |q_{ij}|$.

We now state the main theorem of the paper.

Theorem 4.3. *Let $T_1 = Loop(\mathcal{K}_1, \lambda_1)$, $T_2 = Loop(\mathcal{K}_2, \lambda_2)$, and $T = Loop(\mathcal{K}, \lambda)$. Then T_1 and T_2 implement T if and only if there exists a group homomorphism $h : \pi_1(\mathcal{K}_1) \times \pi_1(\mathcal{K}_2) \to \pi_1(\mathcal{K})$ such that $h([\lambda_1], [\lambda_2]) = [\lambda]$.*

Theorem 4.3 describes only two loop agreement tasks implementing a third, but by finite induction, one can easily generalize this to n tasks. Its proof is broken down into two other theorems, which jointly prove Theorem 4.3. The first theorem is a topological characterization of two tasks implementing a third, while the second theorem is on the correspondence between continuous functions and group homomorphisms.

Theorem 4.4. *Tasks T_1 and T_2 implement T if and only if there exists a continuous function $f : (skel^2(\mathcal{K}_1 \times \mathcal{K}_2), \lambda_1 \star \lambda_2) \to (\mathcal{K}, \lambda)$.*

We prove Theorem 4.4 by proving each direction individually via the following lemmas.

Lemma 4.5. *If there is a continuous function $f : (skel^2(\mathcal{K}_1 \times \mathcal{K}_2), \lambda_1 \star \lambda_2) \to (\mathcal{K}, \lambda)$, then T_1 and T_2 implement T.*

Proof. Suppose such a function f exists, and let Γ_1, Γ_2, and Γ be the specification maps for T_1, T_2, and T, respectively. To prove T_1 and T_2 implement T, we require an $N \in \mathbb{N}$ and a simplicial map $\phi : \mathrm{Bary}^N(\mathrm{skel}^2(\mathcal{K}_1 \times \mathcal{K}_2)) \to \mathcal{K}$ such that for each $\sigma \in \mathcal{I}$, we have $(\phi \circ \mathrm{Bary}^N)(\mathrm{skel}^2(\Gamma_1(\sigma) \times \Gamma_2(\sigma))) \subseteq \Gamma(\sigma)$. We will construct such a ϕ by taking a simplicial approximation of a suitably defined continuous function.

Let p_{01}, p_{12}, and p_{20}, and q_{01}, q_{12}, and q_{20} be the designated edge paths of λ_1 and λ_2, respectively. Consider $X = |(p_{01} \times q_{01})| \cup |(p_{12} \times q_{12})| \cup |(p_{20} \times q_{20})| \subseteq |\mathcal{K}_1 \times \mathcal{K}_2|$ as a topological subspace. Clearly, each $|p_{ij} \times q_{ij}|$ deformation retracts to the corresponding path $|p_{ij} \star q_{ij}|$ in $|\lambda_1 \star \lambda_2|$. In other words, we have a continuous function $H : X \times [0,1] \to |\mathcal{K}_1 \times \mathcal{K}_2|$ such that $H(x,0) = x$, $H(X,1) = |\lambda_1 \star \lambda_2|$, and $H(a,t) = a$ for each $a \in |\lambda_1 \star \lambda_2|$, $x \in X$, and $t \in [0,1]$. Now using Fact 3.11, we can extend H to a continuous function $H' : |\mathcal{K}_1 \times \mathcal{K}_2| \times [0,1] \to |\mathcal{K}_1 \times \mathcal{K}_2|$. In particular, define $r : |\mathcal{K}_1 \times \mathcal{K}_2| \to |\mathcal{K}_1 \times \mathcal{K}_2|$ as $r(x) = H(x,1)$. This is a continuous function from $|\mathcal{K}_1 \times \mathcal{K}_2|$ to itself that fixes $|\lambda_1 \star \lambda_2|$ while collapsing X to $|\lambda_1 \star \lambda_2|$. We restrict r to $|\mathrm{skel}^2(\mathcal{K}_1 \times \mathcal{K}_2)|$ and invoke Fact 3.10 to get a function $g : |\mathrm{skel}^2(\mathcal{K}_1 \times \mathcal{K}_2)| \to |\mathrm{skel}^2(\mathcal{K}_1 \times \mathcal{K}_2)|$ that fixes $|\lambda_1 \star \lambda_2|$ while collapsing $\mathrm{skel}^2(X)$ to $|\lambda_1 \star \lambda_2|$. Now let $F = f \circ g$. This is a continuous function $F : |\mathrm{skel}^2(\mathcal{K}_1 \times \mathcal{K}_2)| \to |\mathcal{K}|$ which maps $\lambda_1 \star \lambda_2$ to λ.

To show F is carried by Γ, first consider the case where $|\sigma| = 1$. Then the point $|\Gamma_1(\sigma) \times \Gamma_2(\sigma)|$ is contained in $|\lambda_1 \star \lambda_2|$, so is fixed under g, and hence mapped to the appropriate point in λ by the given function f. The case $|\sigma| = 2$ is similar. We have $|\Gamma_1(\sigma) \times \Gamma_2(\sigma)| \subseteq X$, which collapses to $|\lambda_1 \star \lambda_2|$ under g. The function f maps this to λ, as desired. The final case is when $|\sigma| = 3$, which does not require any part of the proof above, since $\Gamma(\sigma) = \mathcal{K}$. In all cases, we see that F is carried by Γ. Letting $\phi : \mathrm{Bary}^N(\mathrm{skel}^2(\mathcal{K}_1 \times \mathcal{K}_2)) \to \mathcal{K}$ be a simplicial approximation of F, ϕ is also carried by Γ, so we have the required decision map.

□

Lemma 4.6. *If tasks T_1 and T_2 implement T, then there is a continuous function $f : (skel^2(\mathcal{K}_1 \times \mathcal{K}_2), \lambda_1 \star \lambda_2) \to (\mathcal{K}, \lambda)$.*

Proof. Assuming T_1 and T_2 implement T, we have a simplicial map $\phi : \mathrm{Bary}^N(\mathrm{skel}^2(\mathcal{K}_1 \times \mathcal{K}_2)) \to \mathcal{K}$ that is carried by Γ. In particular, ϕ maps $\lambda_1 \star \lambda_2$ to λ. Let $f : (\mathrm{skel}^2(\mathcal{K}_1 \times \mathcal{K}_2), \lambda_1 \star \lambda_2) \to (\mathcal{K}, \lambda)$, defined by $f(x) = |\phi|(x)$. Then f maps $|\lambda_1 \star \lambda_2|$ to $|\lambda|$ since ϕ does this as well. □

Lemmas 4.5 and 4.6 together prove Theorem 4.4. Next, we prove the correspondence between continuous functions and group homomorphisms. In order to do this, we refer to the following result shown in Herlihy and Rajsbaum [10].

Lemma 4.7. *Let $\mathcal{K}$ and $\mathcal{L}$ be finite, connected, 2-dimensional simplicial complexes, and let $h : \pi_1(\mathcal{K}) \to \pi_1(\mathcal{L})$ be a homomorphism with $h([\sigma]) = [\tau]$. Then there exists a continuous $f : |\mathcal{K}| \to |\mathcal{L}|$ such that $f_* = h$ and $f \circ \sigma = \tau$.*

Theorem 4.8. *There exists a continuous function $f : (skel^2(\mathcal{K}_1 \times \mathcal{K}_2), \lambda_1 \star \lambda_2) \to (\mathcal{K}, \lambda)$ if and only if there exists a group homomorphism $h : \pi_1(\mathcal{K}_1) \times \pi_1(\mathcal{K}_2) \to \pi_1(\mathcal{K})$ such that $h([\lambda_1], [\lambda_2]) = [\lambda]$.*

Proof. First suppose we have a continuous function $f : (\text{skel}^2(|\mathcal{K}_1 \times \mathcal{K}_2|), \lambda_1 \star \lambda_2) \to (\mathcal{K}, \lambda)$. We begin by constructing a homomorphism $h' : \pi_1(|\mathcal{K}_1 \times \mathcal{K}_2|) \to \pi_1(\mathcal{K})$ with $h'([\lambda_1 \star \lambda_2]) = [\lambda]$. Let $\iota : \text{skel}^2(|\mathcal{K}_1 \times \mathcal{K}_2|) \to |\mathcal{K}_1 \times \mathcal{K}_2|$ be the inclusion map, whose induced homomorphism is actually an isomorphism, by Fact 3.9. Then we let $h' = f_* \circ \iota_*^{-1}$. In order to show $h'([\lambda_1 \star \lambda_2]) = [\lambda]$, it suffices to show that $\iota_*^{-1}([\lambda_1 \star \lambda_2]) = [\lambda_1 \star \lambda_2]$. However, notice that $[\lambda_1 \star \lambda_2] = \iota_*([\lambda_1 \star \lambda_2])$ since $\lambda_1 \star \lambda_2$ is already in $\text{skel}^2(|\mathcal{K}_1 \times \mathcal{K}_2|)$, so $\iota_*^{-1}([\lambda_1 \star \lambda_2]) = [\lambda_1 \star \lambda_2]$ as required.

Now, we define the desired homomorphism $h : \pi_1(\mathcal{K}_1) \times \pi_1(\mathcal{K}_2) \to \pi_1(\mathcal{K})$ using h'. Let α_1 and α_2 be loops in $\mathcal{K}_1$ and $\mathcal{K}_2$ respectively. By Fact 3.10, α_1 and α_2 are homotopic to edge loops β_1 and β_2. Now define h as $h([\alpha_1], [\alpha_2]) = h'([\beta_1 \star \beta_2])$. Then it follows that $h([\lambda_1], [\lambda_2]) = [\lambda]$. To show h' is well-defined, we need to show that $|\beta_1 \star \beta_2| \simeq |\beta_1' \star \beta_2'|$ for other edge-loop representatives β_1' and β_2' of α_1 and α_2. We can find edge homotopies H_1 and H_2 taking β_1 and β_2 to β_1' and β_2', respectively, so $H_1 \star H_2$ is an edge homotopy from $|\beta_1 \star \beta_2| \simeq |\beta_1' \star \beta_2'|$, proving that h is well-defined. We have thus found the required h, which proves the forward direction of the theorem.

Now suppose we start with a homomorphism h as described above. We reverse the above argument. We begin by constructing a homomorphism $h' : \pi_1(|\mathcal{K}_1 \times \mathcal{K}_2|) \to \pi_1(\mathcal{K})$. Let α be a loop in $|\mathcal{K}_1 \times \mathcal{K}_2|$. As before, α is homotopic to some edge loop β of $\mathcal{K}_1 \times \mathcal{K}_2$. We define $h'([\alpha]) = h([\rho_1 \circ \beta], [\rho_2 \circ \beta]])$, where the ρ_i are the projection maps. This map is clearly well-defined and a homomorphism since it is the composition of h and the induced maps of the ρ_i.

Now we define a homomorphism $h'' : \pi_1(\text{skel}^2(|\mathcal{K}_1 \times \mathcal{K}_2|)) \to \pi_1(\mathcal{K})$ with $h''([\lambda_1 \star \lambda_2]) = [\lambda]$, using h'. Let ι be the inclusion map, as before. Then we define $h'' = h' \circ \iota_*$. Since $\iota_*([\lambda_1 \star \lambda_2]) = [\lambda_1 \star \lambda_2]$, we see that $h''([\lambda_1 \star \lambda_2]) = [\lambda]$. Finally, we invoke Lemma 4.7 on h'' to obtain the required f. This proves the backward direction of the theorem, and completes the proof.

□

Theorems 4.4 and 4.8 together prove Theorem 4.3.

4.3 Composite Loop Agreement

In defining multiple implementation, we said that tasks T_1 and T_2 implement T if we can use the combined output complex $\text{skel}^2(\mathcal{K}_1 \times \mathcal{K}_2)$ of T_1 and T_2 to solve T. We can think of parallel execution of protocols for T_1 and T_2 as solving a task with input complex Δ^2, output complex

$\text{skel}^2(\mathcal{K}_1 \times \mathcal{K}_2)$, and specification $\Gamma_1 \times \Gamma_2$. We get a task $T' = (\Delta^2, \text{skel}^2(\mathcal{K}_1 \times \mathcal{K}_2), \Gamma_1 \times \Gamma_2)$, and from the definitions it is clear that T_1 and T_2 implement T if and only if T' implements T. Unfortunately, T' is not a loop agreement task, since processes starting on an edge in Δ^2 can land on any edge in $\lambda_1 \times \lambda_2$ and still obey the task specification. However, the subcomplex $\lambda_1 \times \lambda_2$ is not a loop. We address this by defining a loop agreement task $T_1 \times T_2$ with output complex $\text{skel}^2(\mathcal{K}_1 \times \mathcal{K}_2)$ and triangle loop $\lambda_1 \star \lambda_2$. We then show that T' and $T_1 \times T_2$ implement one another, so are equivalent.

Definition 4.9. Let $T_1 = \text{Loop}(\mathcal{K}_1, \lambda_1)$ and $T_2 = \text{Loop}(\mathcal{K}_2, \lambda_2)$ be loop agreement tasks. Then the *composition* of T_1 and T_2, denoted $T_1 \times T_2$, is the loop agreement task $\text{Loop}(\text{skel}^2(\mathcal{K}_1 \times \mathcal{K}_2), \lambda_1 \star \lambda_2)$.

Proposition 4.10. *Tasks T_1 and T_2 implement $T_1 \times T_2$.*

Proof. This is an immediate consequence of Lemma 4.5.

□

Proposition 4.11. *Task $T_1 \times T_2$ implements T_1 (respectively T_2).*

Proof. Lemma 6.2 from Herlihy and Rajsbuam [10] states that it suffices to show there is a continuous function $f : \text{skel}^2(\mathcal{K}_1 \times \mathcal{K}_2) \to \mathcal{K}_1$ mapping $\lambda_1 \star \lambda_2$ to λ_1. It is easy to see that the projection map $\rho_1 : \text{skel}^2(\mathcal{K}_1 \times \mathcal{K}_2) \to \mathcal{K}_1$ satisfies this condition. The proof that $T_1 \times T_2$ implements T_2 is identical.

□

5 Category Theory of Loop Agreement

In this section, we describe a more formal connection between the class of loop agreement tasks and the class of groups, using the language of category theory. We formalize the correspondence between loop agreement tasks and algebraic signatures, and also state one direction of the main theorem using category-theoretic formalism. Intuitively, loop agreement tasks form an organized collection of objects called a "category", with decision maps, or "morphisms", connecting two tasks if one implements the other. The algebraic signature assignment, an example of a "functor" between categories, transforms the loop agreement category into a category of groups. The composition of loop agreement tasks as defined in this paper is actually their "categorical" product.

We begin with some necessary background in category theory; see Mac Lane [14] for a rigorous treatment.

5.1 Categories

A *category* C consists of a collection of *objects*, denoted $\text{Ob}(C)$, and a collection of *morphisms* between those objects, denoted $\text{Hom}(C)$. Each morphism has a *domain* and *codomain*, which are both objects in $\text{Ob}(C)$. If f is a morphism with domain X and codomain Y, then we write $f : X \to Y$. This notation is suggestive of set functions, and indeed the category of sets is a well-known category, and has sets as objects and set functions as morphisms.

As with ordinary functions, morphisms can be composed. Formally, $\mathrm{Hom}(C)$ is equipped with a binary operation called *composition*. If f and g are morphisms, then their composition is denoted $f \circ g$. Note that composition of functions is only defined when the codomain of the first morphism is equal to the domain of the second. Composition is required to be associative; that is, given $f : W \to X$, $g : X \to Y$, and $h : Y \to Z$, we must have $h \circ (g \circ f) = (h \circ g) \circ f$. Composition also requires an identity morphism for each object X, denoted id_X, such that for each $f : X \to Y$, we have $f \circ \mathrm{id}_X = f = \mathrm{id}_Y \circ f$.

As mentioned, sets and set functions comprise the category of sets, denoted **Set**. Another example is the category of topological spaces, where objects are spaces and morphisms are continuous functions between them, and is denoted **Top**. There is also the category of groups, **Grp**, consisting of groups and groups homomorphisms. Algebraic signatures belong to a similar category called the category of *pointed* groups, **pGrp**, whose objects are groups with distinguished elements and whose morphisms are group homomorphisms that preserve distinguished elements.

We say one category C is a *subcategory* of another category C' if the objects and morphisms of C are contained in C'. For example, the category of Abelian groups, **Ab**, is a subcategory of **Grp**. We will also make use of $\mathbf{SimC}_n$, which is the subcategory of **SimC** containing all simplicial complexes of dimension up to n and the morphisms between them.

We can transform objects and morphisms of one category to objects and morphisms of another. Given categories C and D, a *functor* $F : C \to D$ assigns to each object $X \in \mathrm{Ob}(C)$ an object $F(X) \in \mathrm{Ob}(D)$, and to each morphism $f : X \to Y$ a morphism $F(f) : F(X) \to F(Y)$. Functors must respects composition; that is, given two compatible morphisms $f, g \in \mathrm{Hom}(C)$, we must have $F(f \circ g) = F(f) \circ F(g)$. Functors must also respect identity morphisms: $F(\mathrm{id}_X) = \mathrm{id}_{F(X)}$. A common example of a functor is the fundamental group functor $\pi_1 : \mathbf{pTop} \to \mathbf{Grp}$, which maps pointed topological spaces to their respective fundamental groups, and maps continuous functions to their induced homomorphisms. If we consider only path-connected spaces, then π_1 is also a functor from **Top** to **Grp**. The geometric realization $|\cdot| : \mathbf{SimC} \to \mathbf{Top}$ is a functor from the category of simplicial complexes with simplicial maps to **Top**, which maps complexes and simplicial maps to their respective geometric realizations.

We can also combine two objects from a category to produce a new one, which is an operation called the categorical product. The categorical product of two objects is the most general object that maps onto the original two.

Definition 5.1. Let C be a category, and let X_1 and X_2 be objects in this category. The *categorical product* of X_1 and X_2 is the unique object $X_1 \times X_2$ satisfying the following: there exist morphisms (called *projections*) $\rho_1 : X_1 \times X_2 \to X_1$ and $\rho_2 : X_1 \times X_2 \to X_2$ such that for any object X with morphisms $f_1 : X \to X_1$ and $f_2 : X \to X_2$, there exists a unique morphism $f : X \to X_1 \times X_2$ such that $f_1 = \rho_1 \circ f$ and $f_2 = \rho_2 \circ f$. That is, f_1 and f_2 factor through $X_1 \times X_2$ in a unique way, via f. The morphism f is called the *product morphism* of f_1 and f_2.

Examples of categorical products include the product topology for topological spaces [14], the direct product of groups, and the categorical product of simplicial complexes as stated in Definition 3.3 [13].

5.2 The Category of Loop Agreement Tasks

Now that we have the preliminaries of category theory, we define **Loop**, the category of loop agreement tasks. We let Ob(**Loop**) be the collection of all loop agreement tasks $\text{Loop}(\mathcal{K}, \lambda)$, where $\mathcal{K}$ ranges over all finite connected 2-dimensional complexes and λ ranges over all edge loops. Morphisms in **Loop** are valid decision maps between tasks. That is, given tasks $T_1 = \text{Loop}(\mathcal{K}_1, \lambda_1)$ and $T_2 = \text{Loop}(\mathcal{K}_2, \lambda_2)$, a morphism $f : T_1 \to T_2$ is a pair (δ, N) where $N \in \mathbb{N}$ and $\delta : \text{Bary}^N(\mathcal{K}_1) \to \mathcal{K}_2$ is a decision map such that T_1 solves T_2 via δ. Composition of morphisms is defined as follows. Given objects $T_1 = \text{Loop}(\mathcal{K}_1, \lambda_1)$, $T_2 = \text{Loop}(\mathcal{K}_2, \lambda_2)$, $T_3 = \text{Loop}(\mathcal{K}_3, \lambda_3)$, and morphisms $f_1 : T_1 \to T_2$, $f_2 : T_2 \to T_3$ where $f_1 = (\delta_1, N_1)$ and $f_2 = (\delta_2, N_2)$, the composition $f_2 \circ f_1$ is defined as $(\delta_2 \circ \text{Bary}^{N_2}(\delta_1), N_1 + N_2)$. Two morphisms are considered equivalent if their simplicial maps are homotopic[2]. We must now prove that **Loop** is a category.

Theorem 5.2. Loop *is a category.*

Proof. Let T_i and f_i be defined as above, and let Γ_i be the tasks' respective specification maps. To show **Loop** is a category, we need to show that Hom(**Loop**) is closed under composition, composition is associative, and identity morphisms exist. Showing that Hom(**Loop**) is closed under composition amounts to showing that T_1 solves T_3 via $\delta_2 \circ \text{Bary}^{N_2}(\delta_1) : \text{Bary}^{N_1+N_2}(\mathcal{K}_1) \to \mathcal{K}_3$. For brevity we define $\delta = \delta_2 \circ \text{Bary}^{N_2}(\delta_1)$.

From the definition of task implementation, we know that $\delta_1 \circ \text{Bary}^{N_1} \circ \Gamma_1 \subseteq \Gamma_2$ and $\delta_2 \circ \text{Bary}^{N_2} \circ \Gamma_2 \subseteq \Gamma_3$, and we want to show $\delta \circ \text{Bary}^{N_1+N_2} \circ \Gamma_1 \subseteq \Gamma_3$. So $\delta_2 \circ \text{Bary}^{N_2} \circ \delta_1 \circ \text{Bary}^{N_1} \circ \Gamma_1 \subseteq \delta_2 \circ \text{Bary}^{N_2} \circ \Gamma_2 \subseteq \Gamma_3$. We know that $\text{Bary}^{N_2} \circ \delta_1 = \text{Bary}^{N_2}(\delta_1) \circ \text{Bary}^{N_2}$, so $\delta_2 \circ \text{Bary}^{N_2} \circ \delta_1 \circ \text{Bary}^{N_1} \circ \Gamma_1 = \delta_2 \circ \text{Bary}^{N_2}(\delta_1) \circ \text{Bary}^{N_2} \circ \text{Bary}^{N_1} \circ \Gamma_1 = \delta \circ \text{Bary}^{N_1+N_2} \circ \Gamma_1 \subseteq \Gamma_3$. Therefore T_1 solves T_3 via δ, so Hom(**Loop**) is closed under our definition of composition.

Verifying associativity follows a similar argument. Again, let T_i and f_i be defined as above, and in addition let $T_4 = \text{Loop}(\mathcal{K}_4, \lambda_4)$ and let $f_3 : T_3 \to T_4$ with $f_3 = (\delta_3, N_3)$. We must show that $(f_3 \circ f_2) \circ f_1 = f_3 \circ (f_2 \circ f_1)$. But $(f_3 \circ f_2) \circ f_1 = (\delta_3 \circ \text{Bary}^{N_3}(\delta_2), N_2 + N_3) \circ (\delta_1, N_1) = (\delta_3 \circ \text{Bary}^{N_3}(\delta_2) \circ \text{Bary}^{N_2+N_3}(\delta_1), N_1 + N_2 + N_3)$, and $f_3 \circ (f_2 \circ f_1) = (\delta_3, N_3) \circ (\delta_2 \circ \text{Bary}^{N_2}(\delta_1), N_1 + N_2) = (\delta_3 \circ \text{Bary}^{N_3}(\delta_2 \circ \text{Bary}^{N_2}(\delta_1)), N_1 + N_2 + N_3) = (\delta_3 \circ \text{Bary}^{N_3}(\delta_2) \circ \text{Bary}^{N_2+N_3}(\delta_1), N_1 + N_2 + N_3)$, so $(f_3 \circ f_2) \circ f_1 = f_3 \circ (f_2 \circ f_1)$. Therefore composition is associative.

The last requirement, existence of identity morphisms, is trivial to show. Task T_1 solves itself via the decision map $(\text{id}_{\mathcal{K}_1}, 0)$. This finishes the proof that **Loop** is a category. □

Next, we show that the algebraic signature of Herlihy and Rajsbaum can be formulated as a functor between **Loop** and **pGrp**.

Definition 5.3. Let $T_1, T_2 \in \text{Ob}(\textbf{Loop})$ with $T_1 = \text{Loop}(\mathcal{K}_1, \lambda_1)$ and $T_2 = \text{Loop}(\mathcal{K}_2, \lambda_2)$, and let $f_1 : T_1 \to T_2$ with $f_1 = (\delta_1, N_1)$ be a morphism between the two. Then the *algebraic signature functor* is a functor $S : \textbf{Loop} \to \textbf{pGrp}$ defined as follows. Object T_1 is mapped to $(\pi_1(\mathcal{K}_1), [\lambda_1])$, while morphism $f_1 : T_1 \to T_2$ is mapped to $|\delta_1|_* : (\pi_1(\mathcal{K}_1), [\lambda_1]) \to (\pi_2(\mathcal{K}_2), [\lambda_2])$.

[2]By identifying morphisms (in this case homotopic ones), we are constructing a *quotient category* from the original one. In order to construct a quotient category, the equivalence must be compatible with composition. However, it is well known that homotopy is compatible with compositions of continuous functions.

Theorem 5.4. *$S : \mathbf{Loop} \to \mathbf{pGrp}$ is a functor.*

Proof. We use the fact that π_1 and $|\cdot|$ are both functors. We need to show that S preserves identity morphisms and respects composition of morphisms. Let T_1, T_2, and f be defined as above, and let $T_3 = \text{Loop}(\mathcal{K}_3, \lambda_3)$ and let $f_2 : T_2 \to T_3$ with $f_2 = (\delta_2, N_2)$. Then, using the functoriality of π_1 and $|\cdot|$, we have $S(f_2 \circ f_1) = S((\delta_2 \circ \text{Bary}^{N_1}(\delta_1), N_1 + N_2)) = |\delta_2 \circ \text{Bary}^{N_1}(\delta_1)|_* = (|\delta_2| \circ |\text{Bary}^{N_1}(\delta_1)|)_* = |\delta_2|_* \circ |\delta_1|_* = S(f_2) \circ S(f_1)$, so S respects composition. Now let id_{T_1} be the identity morphism of T_1. Then $S(\text{id}_{T_1}) = S((\text{id}_{\mathcal{K}_1}, 0)) = |\text{id}_{\mathcal{K}_1}|_* = \text{id}_{\pi_1(\mathcal{K}_1)}$, so S also preserves identity morphisms. S is well-defined since π_1 cannot distinguish between homotopic functions. We conclude that S is a functor.

□

We are almost ready to prove that composition of loop agreement tasks is in fact the categorical product in **Loop**, but first we need a lemma describing the categorical product in $\mathbf{SimC}_2$, which is slightly different than the one in **SimC**.

Lemma 5.5. *If $\mathcal{K}_1$ and $\mathcal{K}_2$ are objects in $\mathbf{SimC}_2$, then $\text{skel}^2(\mathcal{K}_1 \times \mathcal{K}_2)$ is their categorical product in $\mathbf{SimC}_2$.*

Proof. We first define projection maps $\rho_1 : \text{skel}^2(\mathcal{K}_1 \times \mathcal{K}_2) \to \mathcal{K}_1$ and $\rho_2 : \text{skel}^2(\mathcal{K}_1 \times \mathcal{K}_2) \to \mathcal{K}_2$ as $\rho_1(v_1, v_2) = v_1$ and $\rho_2(v_1, v_2) = v_2$. That is, the ρ_i are the restrictions to the 2-skeleton of the projection maps found in Definition 3.3, so they are clearly simplicial.

Now suppose we have a 2-dimensional complex $\mathcal{K}$ with simplicial maps $\delta_1 : \mathcal{K} \to \mathcal{K}_1$ and $\delta_2 : \mathcal{K} \to \mathcal{K}_2$. Then we define $\delta : \mathcal{K} \to \text{skel}^2(\mathcal{K}_1 \times \mathcal{K}_2)$ as $\delta(v) = (\delta_1(v), \delta_2(v))$. This is the only possible set function δ that makes the diagram commute; that is, δ is the only set function such that $\delta_1 = \rho_1 \circ \delta$ and $\delta_2 = \rho_2 \circ \delta$. This proves uniqueness, but we must also show that δ is simplicial.

Let σ be a simplex in $\text{skel}^2(\mathcal{K}_1 \times \mathcal{K}_2)$. Then $\delta_1(\sigma)$ and $\delta_2(\sigma)$ are simplexes in $\mathcal{K}_1$ and $\mathcal{K}_2$, respectively. But as we have shown, $\delta_1(\sigma) = \rho_1(\delta(\sigma))$ and $\delta_2(\sigma) = \rho_2(\delta(\sigma))$, so in particular, we see that $\rho_1(\delta(\sigma))$ and $\rho_2(\delta(\sigma))$ are simplexes. Hence by Definition 3.3, $\delta(\sigma)$ is a simplex in $\mathcal{K}_1 \times \mathcal{K}_2$, and furthermore it is a simplex in $\text{skel}^2(\mathcal{K}_1 \times \mathcal{K}_2)$ since the dimension of σ is at most 2. So δ is a simplicial map, which proves that $\text{skel}^2(\mathcal{K}_1 \times \mathcal{K}_2)$ is the categorical product of $\mathcal{K}_1$ and $\mathcal{K}_2$ in $\mathbf{SimC}_2$.

□

Note that Lemma 5.5 easily generalizes to $\mathbf{SimC}_n$ and the n-skeleton.

Theorem 5.6. *Composition of loop agreement tasks is the categorical product in* **Loop**.

Proof. Let $T_1 = \text{Loop}(\mathcal{K}_1, \lambda_1)$ and $T_2 = \text{Loop}(\mathcal{K}_2, \lambda_2)$ be tasks as defined before, and let Γ_1 and Γ_2 be their specification maps, respectively. Let $\Gamma_\times$ be the specification map of $T_1 \times T_2$. We must first define decision maps from $T_1 \times T_2$ to T_1 and T_2 that would make $T_1 \times T_2$ the categorical product. We know that $\text{skel}^2(\mathcal{K}_1 \times \mathcal{K}_2)$ is the categorical product of $\mathcal{K}_1$ and $\mathcal{K}_2$ in the category $\mathbf{SimC}_2$, and that the product comes with projection maps $\rho_1 : \text{skel}^2(\mathcal{K}_1 \times \mathcal{K}_2) \to \mathcal{K}_1$ and $\rho_2 : \text{skel}^2(\mathcal{K}_1 \times \mathcal{K}_2) \to \mathcal{K}_2$. Using these, we define maps $g_1 : T_1 \times T_2 \to T_1$ and $g_2 : T_1 \times T_2 \to T_2$ with $g_1 = (\rho_1, 0)$ and $g_2 = (\rho_2, 0)$, and we claim that these maps make $T_1 \times T_2$ the categorical product of T_1 and T_2.

First, we must show that g_1 and g_2 are decision maps solving T_1 and T_2. However, we already showed this in Proposition 4.11. To prove that g_1 and g_2 are the projection maps that make $T_1 \times T_2$ the categorical product, we consider a task T that implements both T_1 and T_2, say via maps $f_1 = (\delta_1, N_1)$ and $f_2 = (\delta_2, N_2)$, respectively. Let $T = \text{Loop}(\mathcal{K}, \lambda)$ and let Γ be its specification map. We must find a decision map that solves $T_1 \times T_2$ from T. Without loss of generality, assume $N_1 \geq N_2$, so let $\delta_2' : \text{Bary}^{N_1}(\mathcal{K}) \to \mathcal{K}_2$ be a simplicial approximation of δ_2. Then $\delta = (\delta_1, \delta_2')$ is a map from $\text{Bary}^{N_1}(\mathcal{K})$ to $\text{skel}^2(\mathcal{K}_1 \times \mathcal{K}_2)$, though it does not necessarily carry λ to $\lambda_1 \star \lambda_2$. Instead, $g = (\delta, N_1)$ is a morphism from $\text{Loop}(\mathcal{K}, \lambda)$ to $\text{Loop}(\text{skel}^2(K_1 \times \mathcal{K}_2), \delta(\lambda))$. However, it is easy to see that $\delta(\lambda)$ is homotopic to $\lambda_1 \star \lambda_2$. Using Fact 3.11, we can extend this to a homotopy on all of $\text{skel}^2(\mathcal{K}_1 \times \mathcal{K}_2)$, so we obtain a continuous function $h : |\text{skel}^2(\mathcal{K}_1 \times \mathcal{K}_2)| \to |\text{skel}^2(\mathcal{K}_1 \times \mathcal{K}_2)|$. Let $\gamma : \text{Bary}^M(\text{skel}^2(\mathcal{K}_1 \times \mathcal{K}_2)) \to \text{skel}^2(\mathcal{K}_1 \times \mathcal{K}_2)$ be a simplicial approximation of h. Then notice that $g' = (\gamma, M)$ is a morphism from $\text{Loop}(\text{skel}^2(\mathcal{K}_1 \times \mathcal{K}_2), \delta(\lambda))$ to $\text{Loop}(\text{skel}^2(\mathcal{K}_1 \times \mathcal{K}_2), \lambda_1 \star \lambda_2)$. So $f = g' \circ g$ is a morphism $f : T \to T_1 \times T_2$. We must also show that $f = (\gamma \circ \text{Bary}^M(\delta), N_1 + M)$ makes the diagram commute. Let $\delta' = \gamma \circ \text{Bary}^M(\delta)$. We know that $\rho_i \circ \delta \simeq \delta_i$ by construction of δ, and it is also clear that $\delta' \simeq \delta$, by construction of δ' and γ. It follows that $\rho_i \circ \delta' \simeq \delta_i$, proving that f makes the diagram commute. Thus we have the required product morphism.

Finally, it remains to show that f is unique. Let f' be any such morphism making the diagram commute, and let δ' be its simplicial map. Then, as set maps, we know that $\delta' = (\rho_1 \circ \delta', \rho_2 \circ \delta')$. However, we are assuming that $|\rho_1 \circ \delta'| \simeq |\delta_1|$ and $|\rho_2 \circ \delta'| \simeq |\delta_2|$, so this allows us to conclude that $|\delta'| = (|\rho_1 \circ \delta'|, |\rho_2 \circ \delta'|) \simeq (|\delta_1|, |\delta_2|)$. Therefore $|\delta'| \simeq (|\delta_1|, |\delta_2|)$, which is homotopic to the map constructed in the existence proof above. So δ is unique up to homotopy, meaning that f is unique. This proves that g_1 and g_2 are satisfactory projection maps, proving that $T_1 \times T_2$ is in fact the categorical product of T_1 and T_2.

□

The category **pGrp** also has products. We define this product, and state without proof that it is indeed the categorical product. This follows immediately from the fact that the direct product of groups is the categorical product in **Grp** [14].

Fact 5.7. *Let (G_1, g_1) and (G_2, g_2) be objects in* **pGrp**. *Then $(G_1 \times G_2, (g_1, g_2))$ is their categorical product.*

Having defined the categorical products in **Loop** and **pGrp**, and together with Theorem 4.3, the next corollary is a simple consequence.

Corollary 5.8. *The functor $S : \textbf{Loop} \to \textbf{pGrp}$ preserves products.*

Proof. Let $T_1 = \text{Loop}(\mathcal{K}_1, \lambda_1)$ and $T_2 = \text{Loop}(\mathcal{K}_2, \lambda_2)$ be objects in **Loop**. Then $S(T_1) = (\pi_1(\mathcal{K}_1), [\lambda_1])$ and $S(T_2) = (\pi_1(\mathcal{K}_1), [\lambda_2])$, so $S(T_1) \times S(T_2) = (\pi_1(\mathcal{K}_1) \times \pi_2(\mathcal{K}_2), ([\lambda_1], [\lambda_2]))$. However, from the proof of Theorem 4.8, we see that $(\pi_1(\mathcal{K}_1) \times \pi_2(\mathcal{K}_2), ([\lambda_1], [\lambda_2])) \cong (\pi_1(\text{skel}^2(\mathcal{K}_1 \times \mathcal{K}_2)), [\lambda_1 \star \lambda_2]) = S(T_1 \times T_2)$, so in fact $S(T_1 \times T_2) \cong S(T_1) \times S(T_2)$. Therefore S preserves products.

□

6 Applications

In this section we present some simple applications of the correspondence between compositions of loop agreement tasks and the products of their algebraic signatures.

Proposition 6.1. *Let T be $(3,2)$-set agreement, and let T' be any other loop agreement task. Then $T \times T'$ and T are equivalent.*

Proof. Recall that $(3,2)$-set agreement is the task $\text{Loop}(\text{skel}^1(\Delta^2), \zeta))$, where ζ is the triangle loop $(0,1,2,((0,1)),((1,2)),((2,0)))$. This triangle loop generates $\pi_1(\text{skel}^1(\Delta^2))$, so $S(T) = (\pi_1(\text{skel}^1(\Delta^2)), [\zeta]) \cong (\mathbb{Z}, 1)$. Let $S(T') = (G, g)$. Then by Corollary 5.8, $S(T \times T') = S(T) \times S(T') = (\mathbb{Z} \times G, (1, g))$. The homomorphism $\phi : \mathbb{Z} \times G \to \mathbb{Z}$ defined by projection onto the first coordinate sends $(1, g)$ to 1, and the homomorphism $\psi : \mathbb{Z} \to \mathbb{Z} \times G$ defined by $\psi(n) = (n, g)$ sends 1 to $(1, g)$. So $T \times T'$ and T implement one another, so are equivalent. □

Since $(3,2)$-set agreement was shown to be universal for loop agreement by Herlihy and Rajsbaum [10], it is operationally intuitive that composing it with any other loop agreement task should not change its relative power.

Proposition 6.2. *Let T be any simplex agreement task, and let T' be any other loop agreement task. Then $T \times T'$ and T' are equivalent.*

Proof. Since the output complex if T is a subdivided simplex, it has trivial fundamental group, so $S(T) = (1, e)$. As before, let $S(T') = (G, g)$. By Corollary 5.8, $S(T \times T') = S(T) \times S(T') = (1 \times G, (g, e))$, which is clearly isomorphic to (G, g). So $T \times T'$ and T implement one another, so are equivalent.

□

Herlihy and Rajsbaum also showed that simplex agreement is implemented from any loop agreement task [10], so it is also intuitively clear that composing a task with simplex agreement should not change the relative power of the original task.

Proposition 6.3. *Let T be any loop agreement task. Then $T \times T$ and T are equivalent.*

Proof. Let $S(T) = (G, g)$. Then by Corollary 5.8, $S(T \times T) = S(T) \times S(T) = (G \times G, (g, g))$. Letting $\phi : G \to G \times G$ be the diagonal map $\phi(x) = (x, x)$, ϕ maps g to (g, g), and letting $\psi : G \times G \to G$ be projection onto a coordinate, ψ maps (g, g) to g. So $T \times T$ and T are equivalent.

□

The above result states that composing a loop agreement task with copies of itself will not change its relative power.

7 Conclusions

It is a common technique to study a class of objects by mapping these objects into a class of simpler ones in such a way that preserves enough information about the original class of objects. This was the idea behind the fundamental group from algebraic topology, and was also the idea of the algebraic signature of Herlihy and Rajsbaum in their work on loop agreement. In this work we formalized and further extended the algebraic signature characterization by defining the composition of tasks and relating compositions of tasks to products of groups, and in doing so we partially answered the questions raised in the original paper. How much further can this characterization be extended; what more can we learn from the algebraic signature functor between loop agreement tasks and groups with distinguished elements? Does this functor have an adjunction?

The categorical techniques in this paper can be applied to general tasks. For example, tasks with decision maps form a category **Task**, with loop agreement as a subcategory. In the case of loop agreement, we are able to extract valuable information about tasks by mapping them into groups. What kind of functors may we apply to general tasks? Also in the case of loop agreement, we were able to identify parallel composition with the category product. Can parallel composition be defined for more general tasks, for instance via $\mathrm{skel}^n(\mathcal{O}_1 \times \mathcal{O}_2)$, and what is its precise operational meaning of parallel composition for general tasks?

References

[1] H. Attiya, A. Bar-Noy, D. Dolev, D. Peleg, and R. Reischuk. Renaming in an asynchronous environment. *J. ACM*, 37(3):524–548, 1990.

[2] S. Chaudhuri. More choices allow more faults: Set consensus problems in totally asynchronous systems. *Information and Computation*, 105(1):132–158, 1993.

[3] M. J. Fischer, N. A. Lynch, and M. S. Paterson. Impossibility of distributed consensus with one faulty process. *J. ACM*, 32(2):374–382, 1985.

[4] E. Gafni and E. Koutsoupias. Three-processor tasks are undecidable. *SIAM J. Comput.*, 28(3):970–983, 1999.

[5] A. Hatcher. *Algebraic Topology*. Cambridge University Press, 2002.

[6] M. P. Herlihy, D. N. Kozlov, and S. Rajsbaum. *Distributed Computing Through Combinatorial Topology*. Morgan Kaufmann, 2013.

[7] M. P. Herlihy and S. Rajsbaum. Set consensus using arbitrary objects. In *Proceedings of the Thirteenth Annual ACM Symposium on Principles of Distributed Computing*, PODC '94, pages 324–333, 1994.

[8] M. P. Herlihy and S. Rajsbaum. Algebraic spans. In *Proceedings of the Fourteenth Annual ACM Symposium on Principles of Distributed Computing*, PODC '95, pages 90–99, 1995.

[9] M. P. Herlihy and S. Rajsbaum. The decidability of distributed decision tasks. In *Proceedings of the Twenty-ninth Annual ACM Symposium on Theory of Computing*, STOC '97, pages 589–598, 1997.

[10] M. P. Herlihy and S. Rajsbaum. A classification of wait-free loop agreement tasks. *Theor. Comput. Sci.*, 291(1):55–77, 2003.

[11] M. P. Herlihy and N. Shavit. The asynchronous computability theorem for t-resilient tasks. In *Proceedings of the Twenty-ninth Annual ACM Symposium on Theory of Computing*, STOC '93, pages 111–120, 1993.

[12] M. P. Herlihy and N. Shavit. A simple constructive computability theorem for wait-free computation. In *Proceedings of the Twenty-sixth Annual ACM Symposium on Theory of Computing*, STOC '94, pages 243–252, 1994.

[13] D. N. Kozlov. *Combinatorial Algebraic Topology*, volume 21 of *Algorithms and computation in mathematics*. Springer, 2008.

[14] S. Mac Lane. *Categories for the Working Mathematician*. Springer Verlag, 1998.

[15] X. Liu, J. Pu, and J. Pan. A classification of degenerate loop agreement. In *Fifth IFIP International Conference On Theoretical Computer Science*, volume 273 of *IFIP*, pages 203–213. Springer, 2008.

[16] X. Liu, Z. Xu, and J. Pan. Classifying rendezvous tasks of arbitrary dimension. *Theor. Comput. Sci.*, 410(21-23):2162–2173, 2009.